

## IN THE SPECIFICATION

Please amend the paragraphs as follows.

On page 18, beginning on line 8.

A<sub>1</sub> | In the preferred embodiment, the data and parity are migrated from the arrangement of Fig.2 to the arrangement of ~~Fig.3~~Fig. 4 such that the data and parity are moved to the position that they would have occupied had the array originally been created with 4 disks. In the preferred embodiment this migration happens gradually, alongside and as part of the normal IO workload of the disks with little impact. The algorithm for this task is as follows:

On page 18, beginning on line 17.

A<sub>2</sub> | A bitmap is initialised which has a single bit to represent each stripe in the array. The bit value indicates whether the data and parity in the strip are in the position shown in Fig.2 or the position shown in ~~Fig. 4~~Fig. 3.

On page 18, beginning on line 25.

A<sub>3</sub> | Reads/Cache stages: If the strip being accessed is one for which the position of the data is different in ~~Fig. 4~~Fig. 3 from the position in Fig. 2 then the bitmap is examined in order to determine which disk to access.

On page 19, beginning on line 7.

A<sub>4</sub> | In an alternative arrangement, a background process is set up to migrate the data and parity from Fig. 2 format to ~~Fig. 4~~Fig. 3 format. This is not preferred since it is much easier to migrate the data and parity between the formats during a full stripe write (no locking or reading needs to be performed) and because the skewing of the load on the disk has only very slight impact for the read only workload which is applied to the disks between segment destages in LSA.

On page 19, beginning on line 29.

A<sub>5</sub> | The benefit of moving the parity to the Fig. 4~~Fig. 3~~ format is that the optimal rotation of parity on the disks is maintained. This is traded off against the cost of relocating the parity.